https://github.com/IoanaDudas/LFTC

SymbolTable class

The SymbolTable class is implemented using a *hash table*, with *linear probing* as a conflict resolution method.
The hash function takes the sum of the ASCII values of each character of the key and applies modulo by the capacity of the table.

Search

Searches a given key in the symbol table and it returns the *index* (hashValue) of the given key, if found or *-1*, if the key is not found in the symbol table.

Input: String key Output: Int

Add

Adds a given key in the symbol table and it returns *True*, if the key is added successfully or *False* otherwise.

Input: String key Output: Boolean

LanguageSpecification class

The LanguageSpecification class is implemented using a *hash map*, where the key is defined as each token (String) and the value is a unique code for the token (Integer), starting with 0 for identifier, 1 for constant and being incremented for each separator, operator and reserved word.

ProgramInternalForm class

The ProgramInternalForm class is implemented using a *hash map*, where the key represents the code of each token, defined in LanguageSpecification class (Integer) and the value is defined as the position in SymbolTable for identifiers and constants or -1 for separators, operators and reserved words.

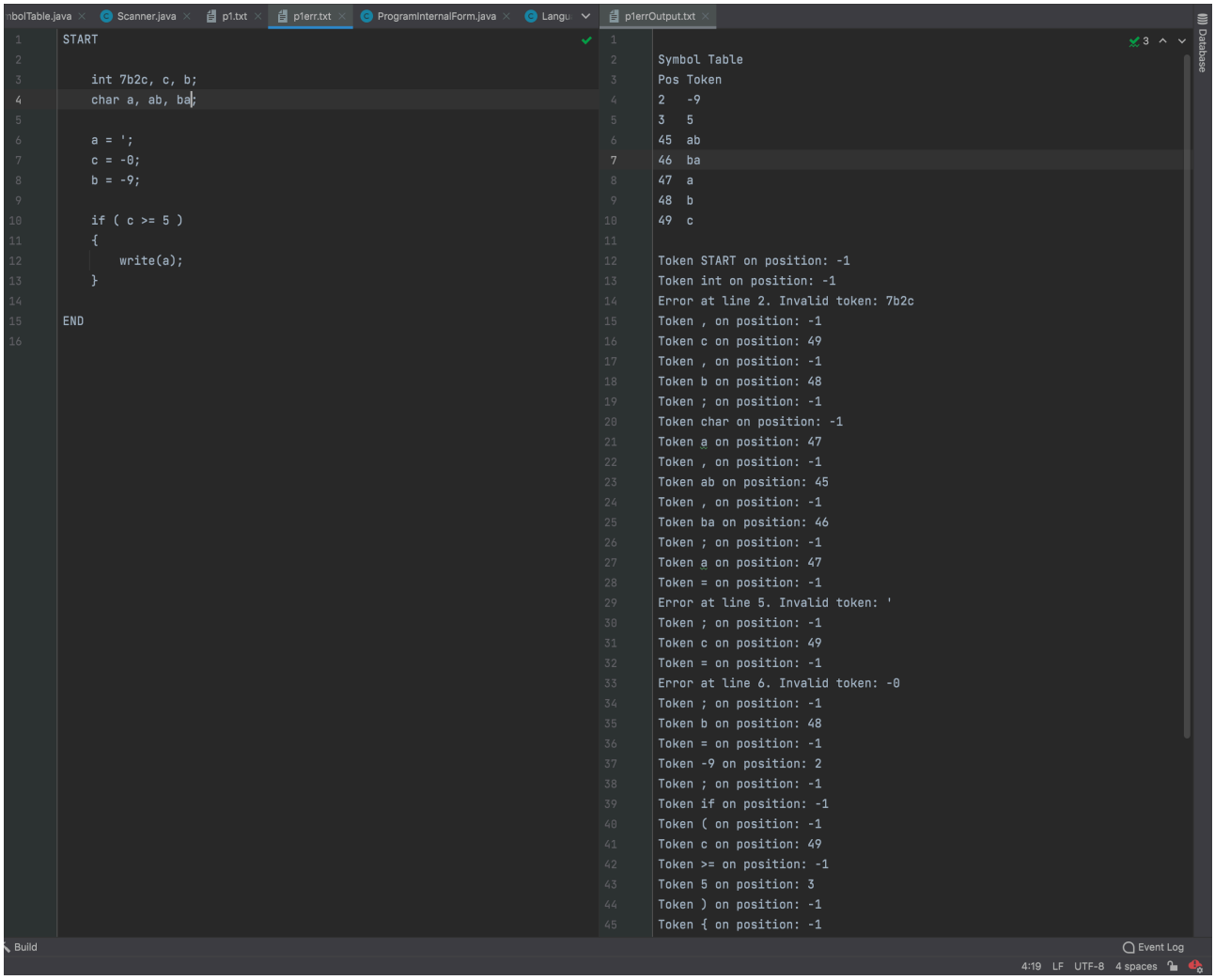
Scanner class

The algorithm splits each line of the program into tokens and for each token: if it is a constant or identifier, it looks up for its position in SymbolTable, and if it is a separator, operator or reserved word, the position will be -1.

Everything will be added to the PIF with the corresponding code from the LanguageSpecification and the determined position in SymbolTable or -1.

The token and the position will be appended to the StringBuilder for better visibility in the output, as well as each lexical error, the line where the error is and the token which caused the error.

Tests



Class Diagram